Building Quality In – The Testing Organization’s Perspective

Yuval Yeret, SPCT
AgileSparks
Limit Defects in Process – Early Quality is Cheap Quality
“We used to feel comfortable to release quarterly and anxious to release every two weeks. “

“Now releasing every two weeks is natural. We are anxious when we can’t do it [due to holiday freeze]”

FiftyOne.com
But how does the typical Testing organization leader feel when seeing the Continuous Delivery Pipeline vision?
See Kent Beck’s idea as described by Markus Gartner at http://www.shino.de/2010/11/04/software-g-forces-the-effects-of-acceleration/
In real life – Agile Testing at the Story level, Waterfall Testing at the Release level. Continuous Integration a tough challenge - Continuous Delivery a faraway dream

"Iteration is too short for everything we need to achieve DONE"

"Let's leave the serious testing for the IP iteration"

The 2-level Test Strategy Pyramid – Story + Release-level
The result - only a limited amount of feedback is early and effective

- A major amount of issues are discovered only in the “end game”
- Finding defects late increases the cost to fix them exponentially
- The result – the need to do a long “end game” to accommodate fixing them, retesting, and even that turns out not to be enough in some cases when the complexity of the big bang “end game” is overwhelming.
Self-Assess – Continuous Testing Dimension

• Individually assess where you are:
  1. Sit – All testing happens towards end of release.
  2. Crawl – Some testing happens by Testing team in follow up sprints. Most still towards end of release.
  3. Walk – Majority testing happens inside the Agile teams. Considerable amount still happens before release.
  4. Run – Minimal deployment/release-testing is needed – IP iteration isn’t heavily used for Testing/Hardening
  5. Fly – All testing is run continuously. System is always ready for deployment/release.

Place a dot on the handout
SAFe Lean-Agile Principles

#1 - Take an economic view
#2 - Apply systems thinking
#3 - Assume variability; preserve options
#4 - Build incrementally with fast, integrated learning cycles
#5 - Base milestones on objective evaluation of working systems
#6 - Visualize and limit WIP, reduce batch sizes, and manage queue lengths
#7 - Apply cadence, synchronize with cross-domain planning
#8 - Unlock the intrinsic motivation of knowledge workers
#9 - Decentralize decision-making

Which SAFe Lean/Agoile Principles are we struggling with here?
First step forward - Achieve Agile Testing Flow at the Story AND Feature/Epic/Iteration level

1. Add Feature/Iteration level testing. Avoid the long wait while minimizing costs

The 3-level Test Strategy Pyramid - Story, Feature/Iteration, Release

2. Left-shift more and more testing through automation, enabling teams using environments/tools/knowhow and more.
Optimize Testing Batch Sizes

• For each testing type:
  – Create the economic batch size tradeoff curve
  – Figure out Ideal batch size without further investment
  – Identify enablers that would enable shifting batch size to the left and improving overall economics

Learn more -
The Test Automation Pyramid

- **Manual**

- **UI**
  - Most expensive automation to develop, run & maintain, so minimize!!!
  - Move majority of E2E testing coverage to Service/API layer
  - QTP/UFT/Selenium/PerfectoMobile/etc.

- **Acceptance (Service/API)**
  - “The Workhorse” of enterprise agile testing
  - Created by testers & developers on agile teams supported by frameworks/guidance by Automation CoE
  - soapUI, etc.

- **Unit Testing**
  - Leverage Agile Teams developer testing to reduce coverage needs
  - Ability to automatically detect (through coverage tools etc.) what is covered

http://www.mountaingoatsoftware.com/blog/the-forgotten-layer-of-the-test-automation-pyramid
Self-Assess – Test Automation Pyramid

• Individually assess where you are:
  1. **Sit** – Most regression testing is actually manual. There’s some UI based coverage but can’t keep up.
  2. **Crawl** – Majority of the regression test suite is based on UI-level automation. It is hard to maintain, slow-running, and brittle – often fails without reason and requires lengthy analysis. (Semi-Automated)
  3. **Walk** – Inverted pyramid – Developers are writing unit tests for new code that reduces some need for other coverage for that code, Some API testing exists, Still a lot of UI based coverage. Some manual testing is needed beyond the automation pyramid.
  4. **Run** – Tube – Similar amount of coverage coming from each area. Minimal manual testing still needed.
  5. **Fly** – Upside pyramid with no manual regression testing required and fully automatic and reliable continuous testing integrated into continuous integration.

Place a dot on the handout
In classic testing – we find defects to assure quality

Classic agile development – Test Design in parallel or after coding

Test automation - Happens towards the end of the agile iteration or is left over for next iterations / automation team.
Shift even more to the left... – Specify Acceptance Tests FIRST to DRIVE Design/Development

ATDD = Acceptance Test Driven Development - Build Quality Into Design – preventing defects rather than just finding them (a.k.a BDD – Behavior Driven Development)

1. ATDD Thinking
   Use test scenarios to guide design

Test Automation

http://www.agilesparks.com/test-first-reading-list
Specify/Groom using Acceptance Tests

Backlog Grooming / Acceptance-Tests Specification Workshop

- Identify acceptance criteria/tests for the next stories in the backlog.
- Use acceptance tests as guidance for slicing stories smaller to enable more effective agile collaboration.
Team pulls a few stories

Testers create more detailed test examples
Developers use them to guide technical design/specification
Story is pulled into development

Testers focus on detailed test plans, automation preparation, data preparation.
Developers code and do developer/unit/small testing
Development done means the acceptance tests already pass

Result: Teams using ATDD get much higher, automated, test coverage (some above 90%) than the typical test-last teams. Both due to discipline as well as due to developers awareness to the acceptance tests.

Developer verifies the story passes the team “ready for story testing” criteria:

- Code is “in the build”
- The acceptance tests for this story pass in the dev env
- Good code coverage using automated unit tests
- Sanity test passes

Developer marks story as dev done, ready for testing
Developers meanwhile pull more stories into coding.

Tester pulls story into testing:

- Tester finds significantly fewer defects
- Product Owner is happier with how the story works & finds fewer surprises.
- Both as a result of deeper alignment achieved in the specification workshops with ATDD scenarios.
Self-Assess – Test-First Dimension

• Individually assess where you are:
  1. **Sit** – Test design, execution, and automation happens after software is designed, when interfaces are stable. Story/Iteration Definition of Done doesn’t include testing.
  2. **Crawl** – Test design and execution is included in definition of Done and executed by the Agile team – test design for each story happens after the story is build coded.
  3. **Walk** – Test design happens in parallel to defining/building the story. Feedback from test design can sometimes influence software functionality/design choices.
  4. **Run** – Test design happens FIRST and is always used as feedback/guidance when defining the story.
  5. **Fly** – Test design happens FIRST and is used to create mini-stories each focusing on one small slice/scenario that can then be designed and built and tested very quickly by the team enabling fast effective flow.

Place a dot on the handout
All together now—Whole Team Automation guided by ATDD thinking and the Test Automation Pyramid

1. ATDD Thinking
   Use test scenarios to guide design

2. ATDD Automation
   Using BDD tool and the relevant dev-friendly test tools
   Built in parallel to development at the right level (according to the test automation pyramid)
ATDD/BDD Tools Options and Architecture

Act as a layer on top of test drivers that actually “drive” the SUT (System Under Test)

- SoapUI
- Selenium
- QTP
- Mobile
- Web Services
- Web Browser
- Web Browser + Fat Clients
- Other...
Self-Assess – Test-First Tooling Dimension

• Individually assess where you are:
  1. Sit – relying on record&play techniques and cannot support test-first.
  2. Crawl – Developers can automate during/before development using APIs.
  3. Walk – Whole team can automate significant amounts of coverage during/before development using a combination of APIs to the system under test and an ATDD/BDD tool enabling non-developers to create scenarios.
  4. Run – ATDD/BDD tooling runs all functional testing, is integrated into CI, and is also used for some of the non-functional tests.
  5. Fly – ???

Place a dot on the handout
Testers need to provide a different kind of value within an agile team/organization

- Being Champions of the Product and the Customer/User.
- Specializing in Performance/Security/Load/etc.
- Shining light on where to focus quality efforts by analyzing risk probability and Impact.
Agile Tester as Test/QA expert rather than automation engineer

- **QA - Accountable to Quality:**
  By *Enabling* it rather than *Owning* it

- Focus on elaborating scenarios to help guide the team as well as advanced and exploratory tests – especially for the riskier stories

- Automation development and execution – a team responsibility – Developers typically handle the majority of the automation development

- ATDD/BDD tools enable QA people to guide the team and enable automation without being coders themselves.
Most Agile Testers join the Agile teams to help “Build Quality In”. In some cases the System Team takes on deeper more specialized testing tiers – like full coverage of the compatibility matrix.
Self-Assess – The Agile Tester Dimension

• Individually assess where you are:
  1. Sit – Testers are in a different organization on different teams.
  2. Crawl – Teams get allocated “half a tester”
  3. Walk – Agile Teams include testers that focus on that team and can really support Test-First efforts.
  4. Run – Agile Testers cover the majority of testing work including some of the more specialized test types
  5. Fly – Teams look up to their testers to guide the way towards quality – test-first isn’t just a process it is the way the teams think and behave.

Place a dot on the handout
Summary - Navigate the SAFe Implementation Roadmap from the Testing Organization’s perspective

Testing/QA Leader for the organization

All Testers/QA Engineers participate

Champion building quality in!

Test-First Automation + Allocate capacity to close automation gaps

Product Ownership for Test Automation Backlog of Enablers

Inspect and Adapt organizational Quality metrics

Figure out impact on Dev/Test Infra, Quality Practices, Create System Team

Shift more and more testing left, Reduce batch sizes, bring more and more capabilities into the Agile teams
Questions?